

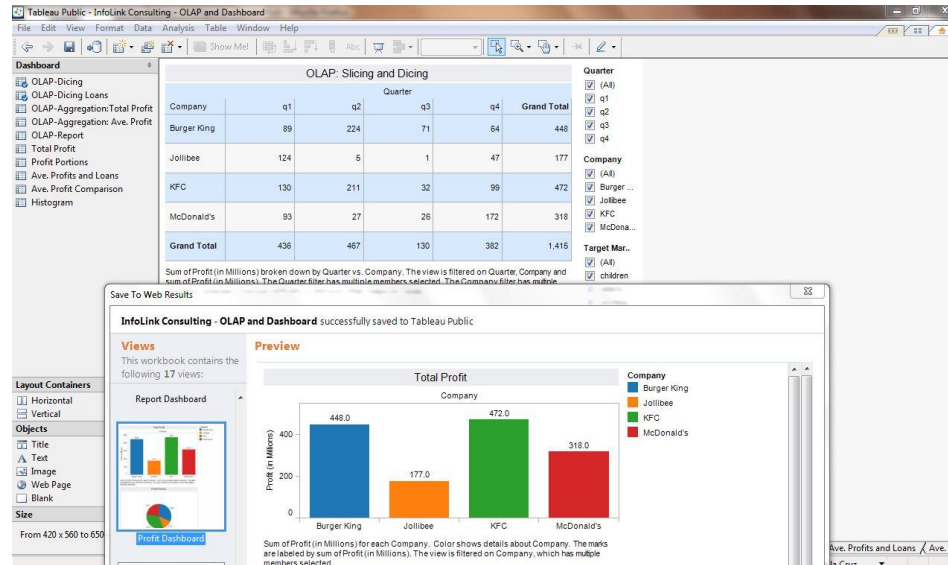
Multidimensional Modelling

Scalable Data Engineering

Motivation

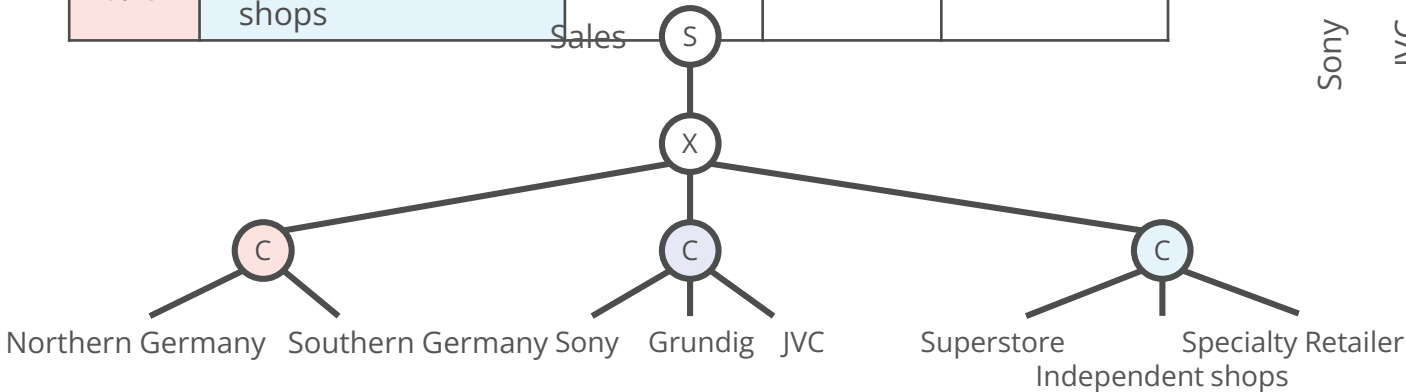
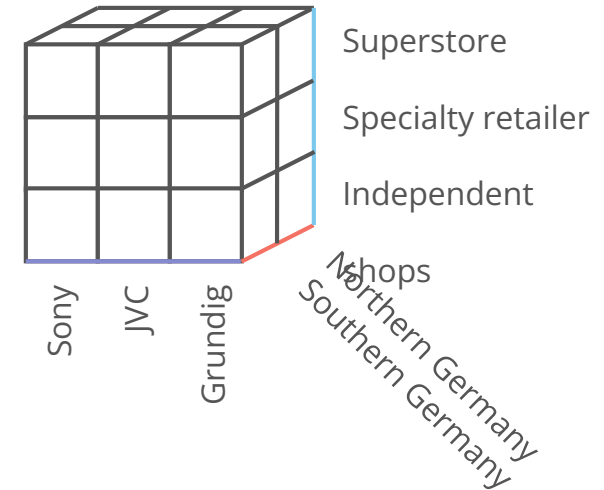
Reporting and interactive Analysis

- OLAP (Online Analytical Processing) on multidimensional data model
- Data Mining: Search for unknown patterns or relations in data
- Visualization



Data Representation

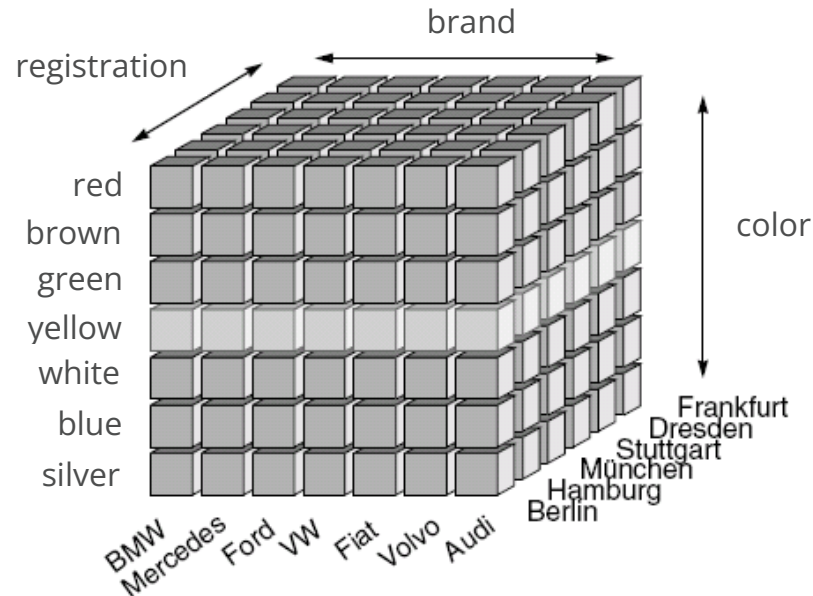
Sales		Sony	JVC	Grundig
Northern Germany	Superstore			
	Specialty retailer			
	Independent shops			
Southern Germany	Superstore			
	Specialty retailer			
	Independent shops			

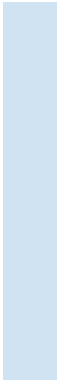


Basic concept

Central data structure: multidimensional cube

- Descriptive data (categorical attributes)
- Quantified data (sum attributes)

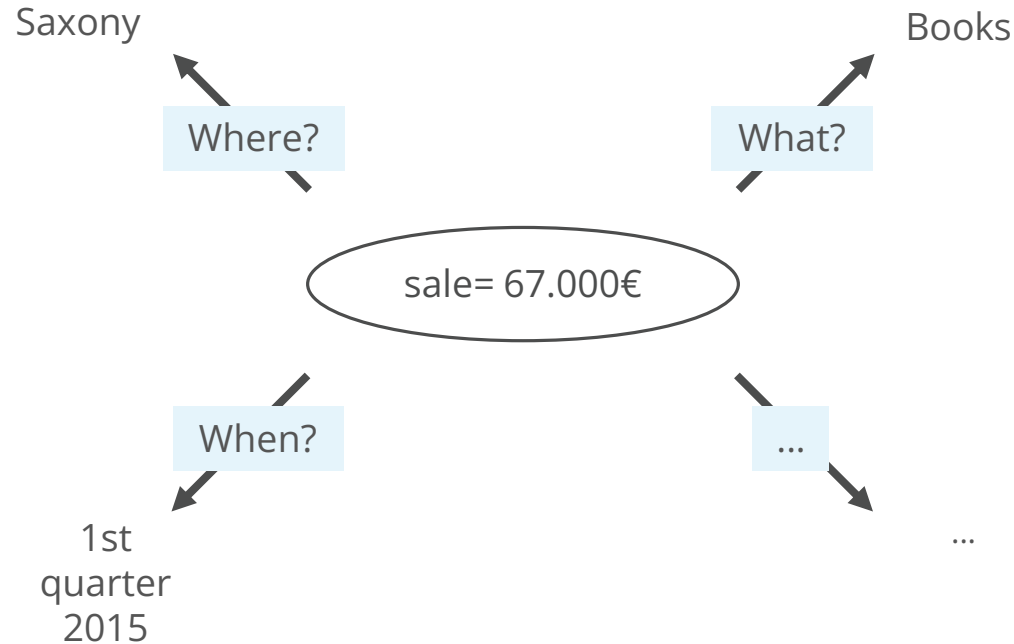




Multidimensional modeling - Basics

Multidimensional data modeling

Dimensions



“Predict” analytic patterns of users

- Drill-paths for navigations operators
- Limit to *meaningful* aggregation options

Data structures

- **Descriptive information** (cube edges) → dimensions
 - Hierarchies, dimensional attributes
 - Structural basis for selection and aggregation
- **Quantified information** (cube cells) → facts
 - Measures / key performance indicators for analysis / aggregation

Goal

- Orthogonal dimensional descriptions
- Clear separation of measures

Multidimensional schema Ω

- **Set of dimension hierarchies** ($D^1, \dots, D^m$)
- **Set of measures** ($M^1, \dots, M^n$)

Dimensions / Dimension hierarchy

- Partially ordered set D of categorical attributes

$$(\{D_1, \dots, D_n, Top_D\}; \rightarrow)$$

- Top_D is a generic maximum element w.r.t. " $\rightarrow$ ", i.e.

$$\forall i (1 \leq i \leq n): D_i \rightarrow Top_D$$

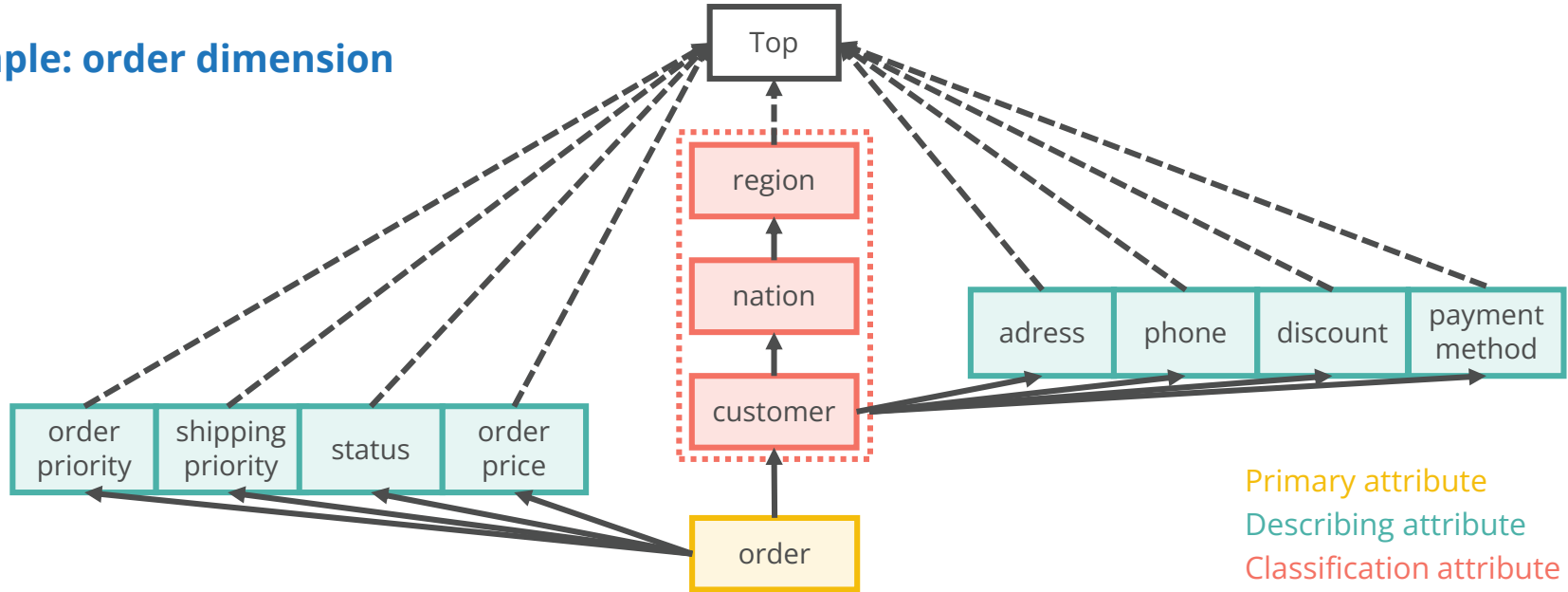
- There is a D_i with finest granularity, i.e. $D_i \rightarrow D_j$ for all D_j
- " $\rightarrow$ " denotes the functional dependency
- Partial ordering allows arbitrary parallel hierarchies

Roles of categorical attributes

- **Primary attribute:** finest granularity (e.g. order)
- **Classification attribute:** implies multi-stage categorization on value level (e.g. customer, nation)
- **Describing attribute:** additional description (e.g. status, phone number)

Schema of a dimension

Example: order dimension



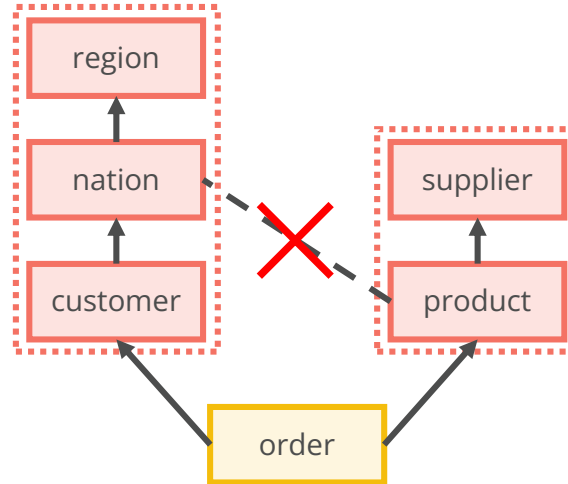
Primary attribute
Describing attribute
Classification attribute

Orthogonality

- No functional dependencies between attributes from different dimensions

Schema of a dimension

Example: multiple hierarchies



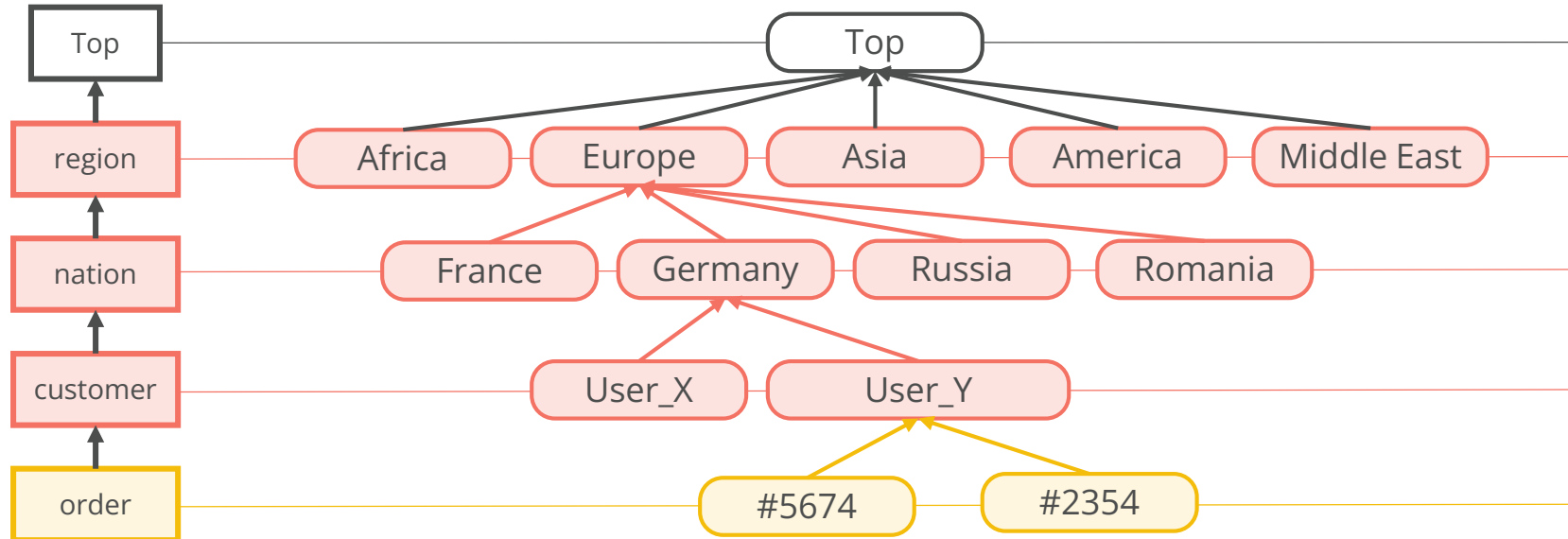
Primary attribute
Describing attribute
Classification attribute

Orthogonality

- No functional dependencies between attributes from different dimensions

Functional dependencies define tree structure on instances

- Functional dependency corresponds to 1:N-relation!
- Every path from a classification attribute to Top defines a classification hierarchy



Quantifying part of a multidimensional data schema

- Facts, (derived) measures

Fact

- Defined as $F = (G, \text{SumType})$
- Granularity: $G = \{G_1, \dots, G_n\}$
 - Subset of all categorical attributes of all existing dimension schemata $DS_1, \dots, DS_n$
- Summation type: $\text{SumType} \in \{\text{FLOW}, \text{STOCK}, \text{VPU}\}$

Example

- Delivered quantity with granularity $G = \{\text{orderNo}, \text{partNo}, \text{supplierNo}\}$

Measure $M = (G, f(F_1, \dots, F_k), \text{SumType})$

- Granularity G
- Calculation formula $f()$
- Non-empty subset of all existing Facts $F_1, \dots, F_k$

Calculation formula $f()$

- Scalar function $+, -, \cdot, /, \text{mod}$ etc.
 - Example: sales tax = quantity $\cdot$ price $\cdot$ tax rate
- Aggregation functions
 - A function $H()$ that summarizes n individual data values with a value derived by a certain function
 - Standard aggregation functions: $\text{SUM}()$, $\text{AVG}()$, $\text{MIN}()$, $\text{MAX}()$, and $\text{COUNT}()$
 - Example: Total sales = $\text{SUM}(\text{quantity} \cdot \text{price})$
- Order-based functions
 - Example: cumulation, top-k calculations

Facts, measures & data cube

Data cube: $C = (DS, M)$

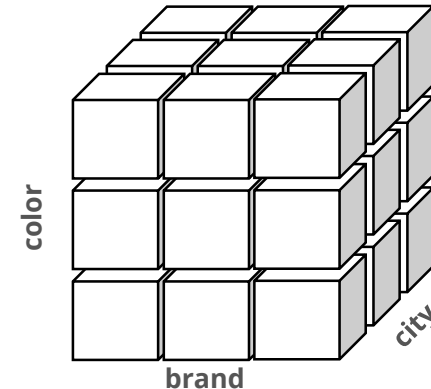
- Set of dimensional schemata: $DS = \{D^1, \dots, D^n\}$
- Set of fact-based measures: $M = \{M^1, \dots, M^m\}$

Domain of a data cube

- Cartesian product of all value ranges of all attributes of the cube schema

Data cube instance

- All cube cells from the cube domain
- Not a subset as in relational model
- Cube is just a metaphor!
 - Almost never, all cells are present (sparsity)
 - Non-existent values on the implementation level become NULL or 0 on the model level!



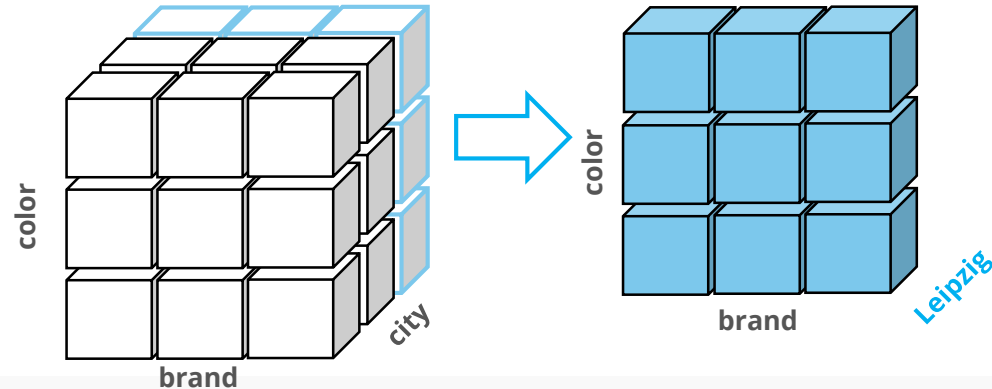
Operations on data cubes

Slicing

- Select “slices of a cube”
- Selection via specification of classification attributes of one dimension
- Changes categorization, not granularity

Example

- city=“Leipzig”



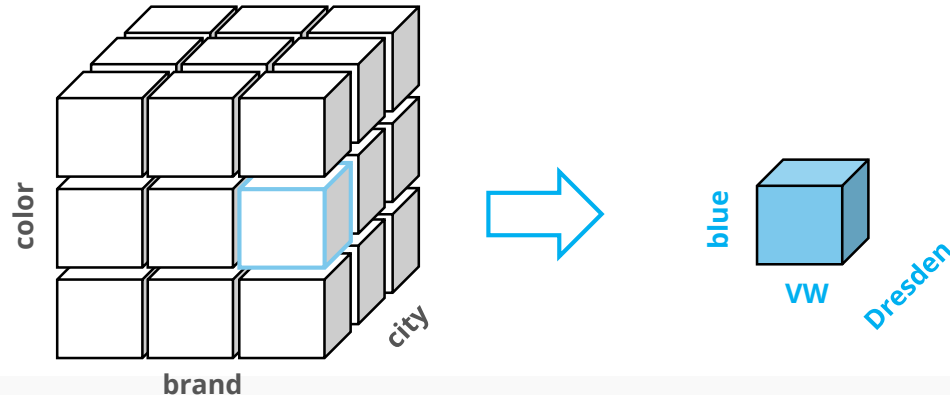
Operations on data cubes

Dicing

- Selection by specification of classification attributes of **multiple** dimensions
- Selects a sub-cube

Example

- city="Dresden", AND
- brand="VW", AND
- color="blue"

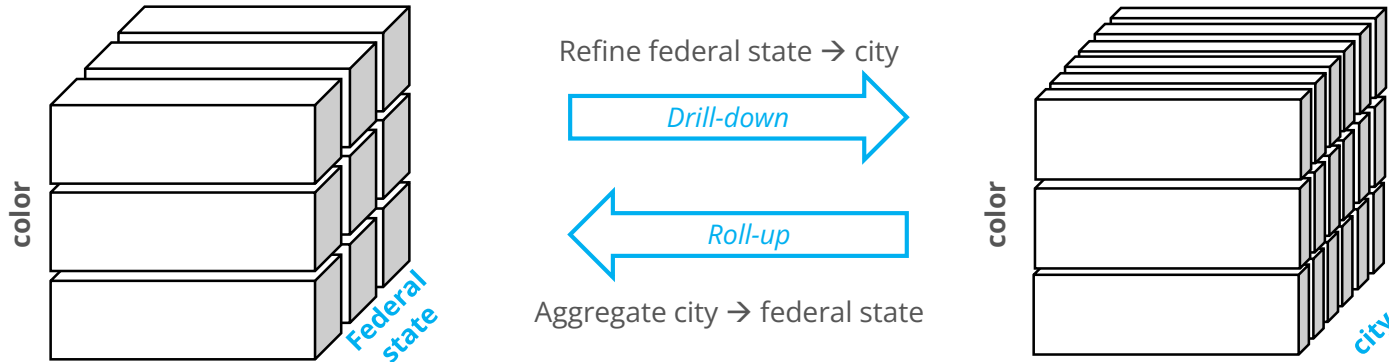


Drill-down / Roll-up

- Moving along the dimension hierarchy
- **Drill-down**: disaggregation of measures into sub-measures
- **Roll-up**: aggregation of measures into super-measures
- Changes granularity, not Categorization

Example

Hierarchy: Region $\leftarrow$ Nation $\leftarrow$ Federal state $\leftarrow$ City

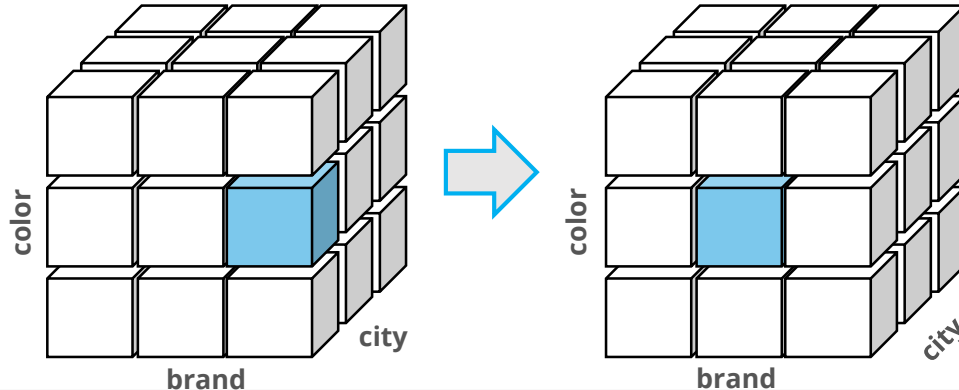


Drill-Across

- Change from one sub-cube to another
- Dimension stays on the same hierarchy-level, but selection value changes
- Also change of data cube (join of multiple data cubes)

Example

- Origin
 - city="Dresden", AND
 - brand="VW", AND
 - color="blue"
- Target
 - city="Dresden", AND
 - brand="Ford", AND
 - color="blue"



Problem

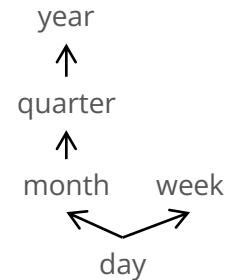
- Not all functions can be summed up (median, quantiles, standard deviation)
- Even simple aggregation functions (SUM, AVG, MIN, MAX, COUNT, ...) cannot be aggregated further at will

Change of granularity

- $G = (G_1, \dots, G_n)$ is finer (same) $G' = (G'_1, \dots, G'_k)$, i.e. $G \leq G'$ iff, for each $G'_i \in G'$ exists a $G_i \in G$, s.t. $G_i \rightarrow G'_i$
- Coarsening / refinement of granularity adding / removing categorical attributes
- Example
 - (orderNo, partNo) $\leq$ (customerNo, brand) $\leq$ (market segment)

Necessary characteristics for summability

- Disjointness, completeness, type compatibility



Characteristics for Summability

Disjointness

- An individual value is considered **exactly once** during calculation
- General: summability is given if functional dependencies exist between the categories that should be summed up

Example: statistical table with undergraduate students in basic studies (2 years)

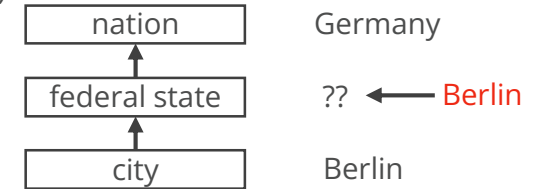
Number of students	2020	2021	2022	Total
Computer science	15	17	13	28
Economics	10	15	11	21
Total	25	32	24	49

- Case 1: sum by year gives correct results
- Case 2: sum up number of computer science students
 - naive: $15 + 17 + 13 = 45$ gives wrong result (2 year overlap)
 - Assume faculty founding in 2020: $15 + (17 - 15) + (13 - 2) = 28$

Characteristics for Summability

Completeness

- Measures on higher aggregation-level are completely derived from measures of lower aggregation-levels
- Possibly, the whole space cannot be captured
 - Example: assume Berlin is not modeled as a federal state (city $\rightarrow$ federal state)
 - Weak functional dependency ($A \Rightarrow B$): for each $a \in A$ exists at most one $b \in B$
 - example: city $\Rightarrow$ federal state
- Remove weak functional dependencies
 - NULL, OTHER, dummy values



Example

- Number of restaurants in the triangle Dresden-Leipzig-Chemnitz
- OTHER for inns, not located in one of the cities

Number of restaurants	2010	2011	2012	2013
Dresden	34	38	31	29
Leipzig	123	121	128	131
Chemnitz	21	18	24	27
OTHER	11	13	13	12
Total	189	190	196	199

Type compatibility – Three classes

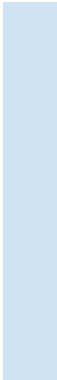
- Flow (event at time T)
 - Can be aggregated freely
 - Examples: order quantity of a certain item per day, number of traffic deaths per month, sales, earnings per year,...
- Stock (status at time T)
 - Can be aggregated freely, without a temporal dimension
 - Items in stock over time → disjointness voided
 - Example: number of school kids per month, city residents, stock, ...
- Value-Per-Unit (VPU)
 - Current states, that cannot be summed
 - Use of COUNT()-, MIN()-, MAX()- und AVG() is allowed
 - Example: value added tax rate, exchange rate, unit costs, ...

Characteristics for Summability

Summary

(part of group by clause)

	FLOW	STOCK: aggregation over temporal dimension?		VPU
		no	yes	
MIN/MAX	✓	✓		✓
SUM	✓	✗	✓	✗
AVG	✓	✓		✓
COUNT	✓	✓		✓



Relational Modelling of MDM

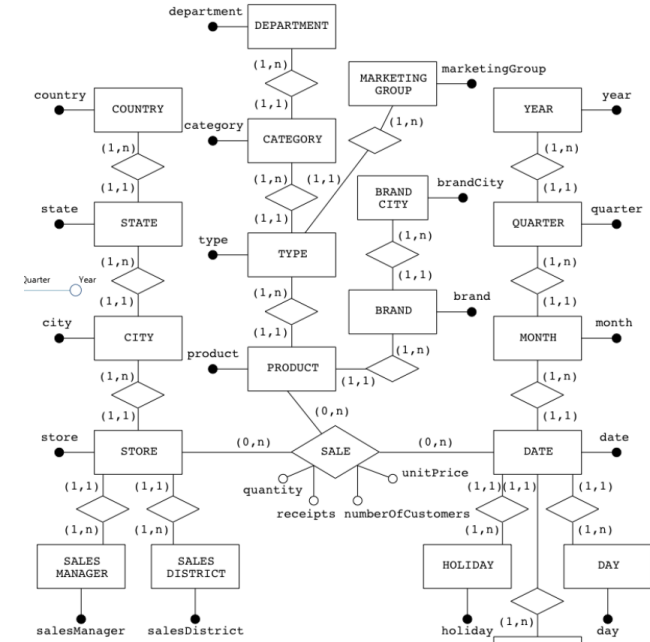
Multidimensional database design

	Classical relational database design	Multidimensional database design	
Conceptual schema (semi-formal)	Variants of entity-relationship modelling	Different modelling languages, e.g. mE/R, mUML, ADAPT,...	
Logical schema (formal)	Relations with attributes	Data cube: facts and measures	
		Dimensional hierarchy with categorical attributes: classifying and describing attributes	
Internal schema	Memory organisation (primary/secondary indexes, partitioning,...)	Relational storage (ROLAP): Star/Snowflake schema patterns	Multidimensional storage (MOLAP): native implementation

dimensional
**attribute **



Multidimensional ER



Relational Data model

- Avoiding *redundancies*
- OLTP queries
- Preferably efficient querying of *individual* entries

In Data Warehouses

- OLAP queries usually affect *multiple* entries
- Value ranges and/or aggregates
- **Redundancy** can shorten query response time

→ *Different relational data models*

Mapping the Multidimensional Model

Multidimensional Model is conceptual not physical!!

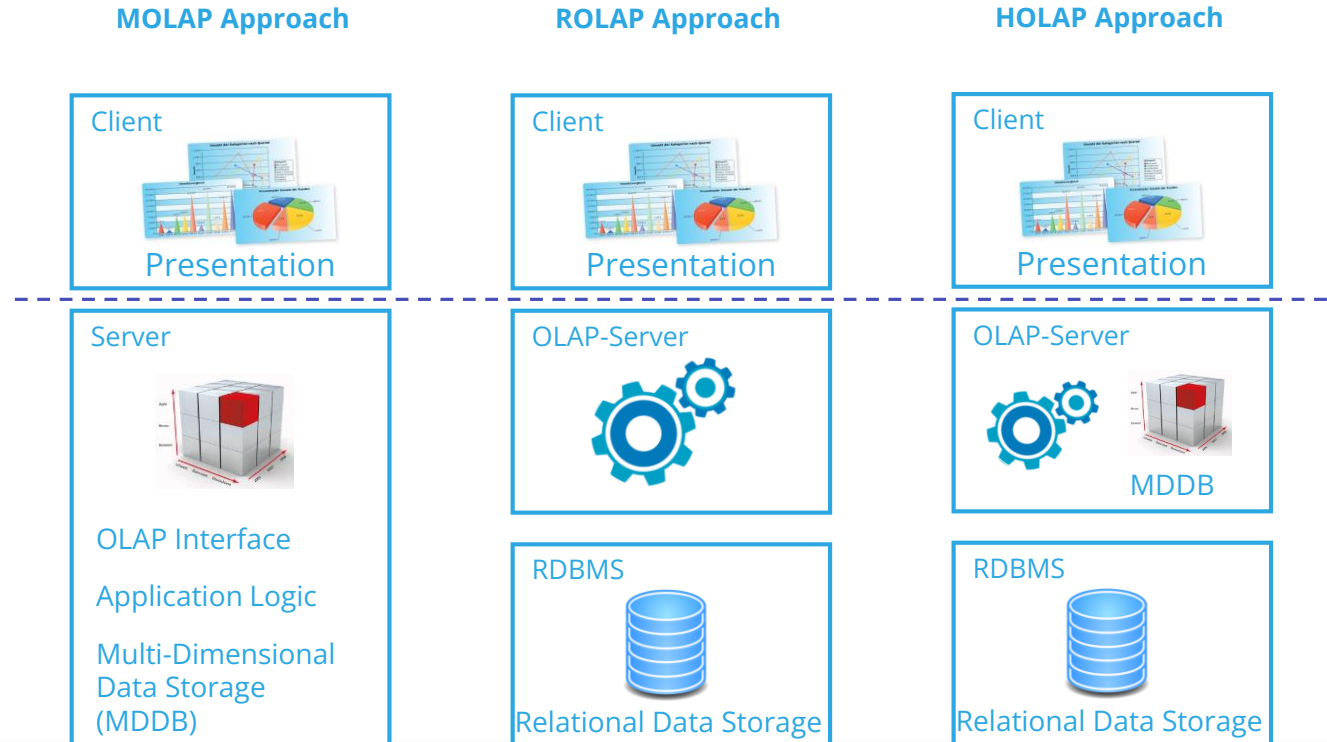
Requirements for physical implementation of the multidimensional Model

- Physical storage of OLAP system data
- High data volume
- Historic data
- Consistently fast response times
- Multi-user operation
- Scalability, maintainability, flexibility, secure data access

Realization w.r.t. analysis and loading phase

- Goal of analysis: efficient extraction of information from data
- Data base for applications and users

Mapping alternatives



Properties

- Raw data & aggregates stored in OLAP server
- Direct storage as cubes

Pros

- No transfer of modeling concepts
- Separation of descriptive und quantified attributes fixed in the model
- Multiple integrated statistical operators (esp. OLAP functions)
- Fast response times through pre-calculation
- Very efficient due to optimized storage structures

Cons

- Proprietary systems
- Often compression is missing (sparsity) , scalability (MB to GB)
- High storage requirements (factor 10 to 100) due to aggregation/pre calculation
- Often limited number of dimensions (10 to 64)

Products

- Oracle Express, Hyperion Essbase, Cognos Powerplay, Seagate Holos

Properties

- Data is stored in RDBMS
- Information is created via SQL
- Additional tables for aggregates

Pros

- Proven database technology
- Scalability for big volume data / high number of dimensions

Cons

- Installation overhead (interaction of different components)
- Overhead for mapping the Data Cube and complex queries to SQL
- Extensive Meta data management for parameters
- Slower

Products

- Almost all RDBMS vendors (Oracle, DB2, Microsoft, etc.), SAP BW / SIMCE

Properties

- „best of both worlds“
- Data partially stored in MOLAP Cubes and RDBMS

Pros

- Proven database technology
- Improved scalability / response times

Cons

- Installation overhead (interaction of different components)
- Complexity / Handling, Which data should be stored where?

Products

- MicroStrategy, Essbase, Mondrian, Microsoft Analysis Services

Multi-Dimensional OLAP (MOLAP)

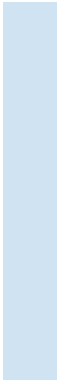
- + Simple Mapping; efficient native multidimensional operations
- Sparsity: typical data cubes populated at 5%
- Scalability: problematic mapping of multidimensional arrays (100s GBs) to storage structures

Relational OLAP (ROLAP)

- + Scalable, proven technology
- Relational data bases are designed for OLTP, not for OLAP
- Missing support of all multidimensional operations (SQL extensions)
- Poor performance (depending on use case), but → column-stores

Hybrid OLAP (HOLAP)

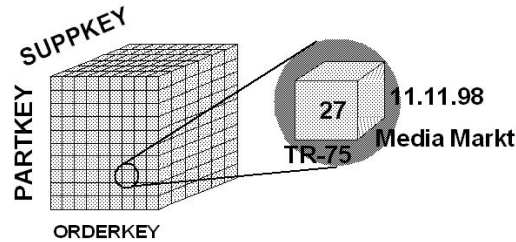
- + Best of both worlds
- System complexity



ROLAP schema alternatives

Relational storage of multidimensional data

- Table with primary key as compound of foreign keys from dimensions
- Only populated cube cells are stored as tuples



ORDERKEY	SUPPKEY	Partkey	QUANTITY
TR-75	MediaMarkt	11.11.98	27
AC300	ProMarkt	18.11.12	5
..	..	..	..
..	..	..	..
..	..	..	..

Separation of structure and content leads to

- A set of dimension tables
- A central fact table
- Star/Snowflake schema

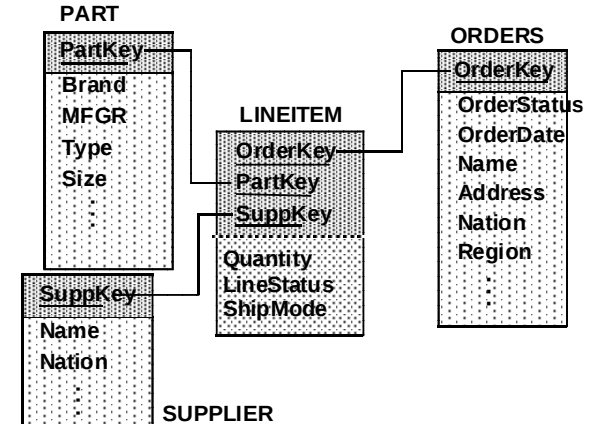
Star / Snowflake Schema

Fact table (content)

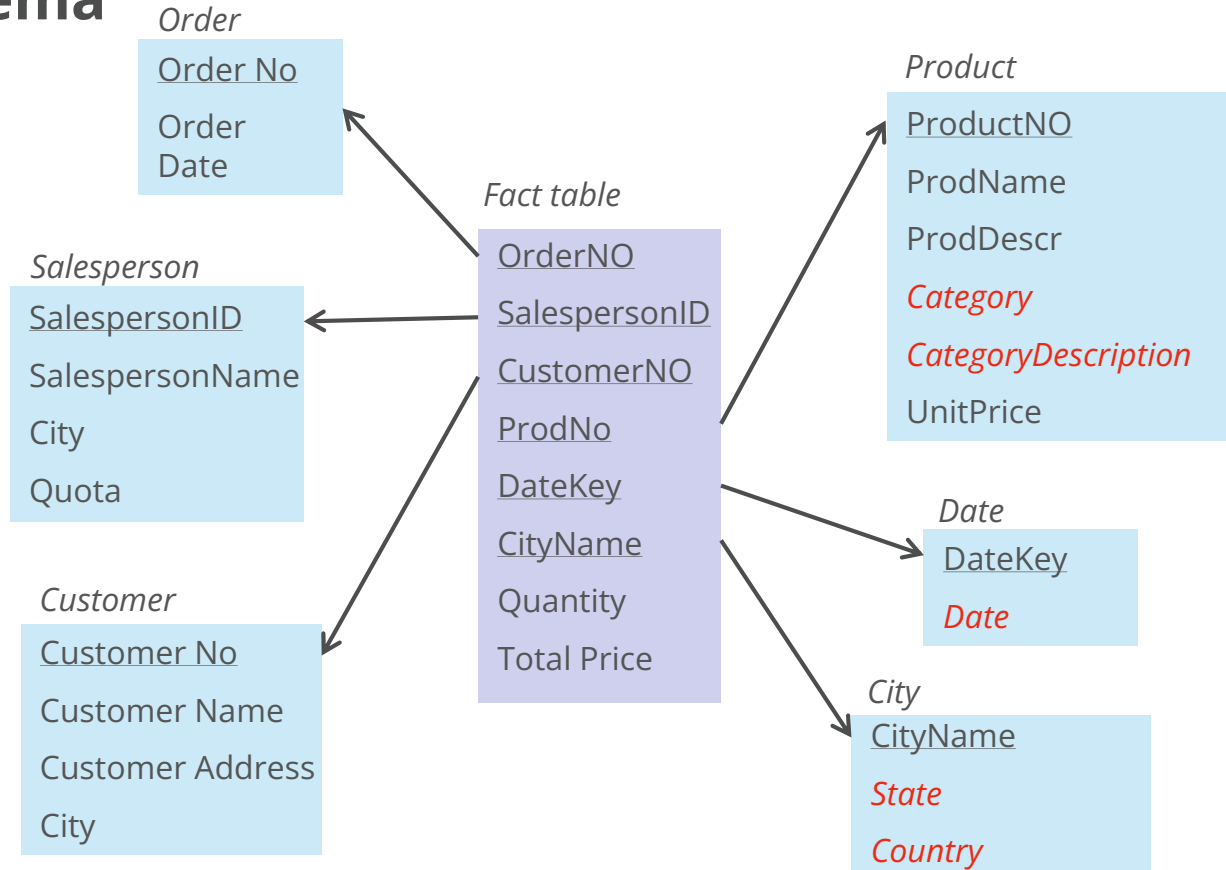
- Central table
- Compound primary key from foreign keys of dimension tables (explicit assignment of facts)
- Few columns, many tuples (millions, billions)
- Typically, only numeric data types

Dimension tables (structure)

- Classifying and descriptive attributes
- Many columns, but few tuples (< 10% of fact table)
- Used for selection (by utilizing bitmap-indices)
- Used for aggregation (group-by-clause)
- Foreign key links dimension table and fact table



Star Schema



Pros

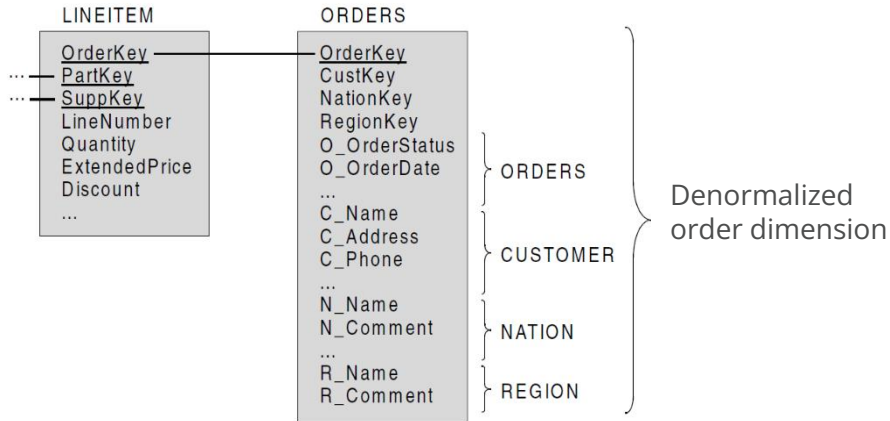
- Intuitive for users
- Low response time because of read-optimization
- Adaptability to changing structure
- Partitioning and optimization possible
- Less redundancy

Cons

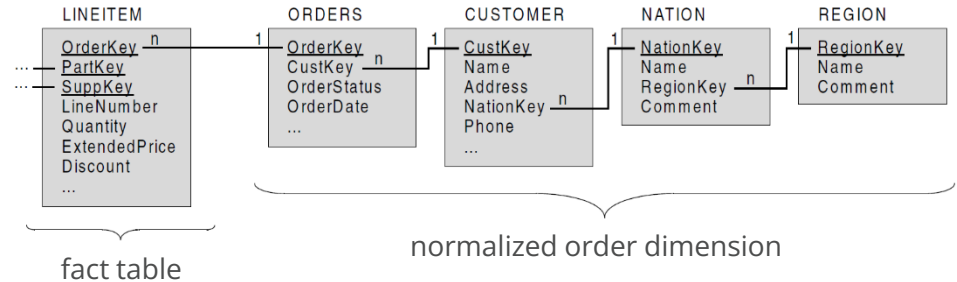
- Not that update friendly
- Multi-hierarchies can be only modeled indirectly
- No distinction between classifying and descriptive attributes (only implicit hierarchies)

Comparison of Star and Snowflake

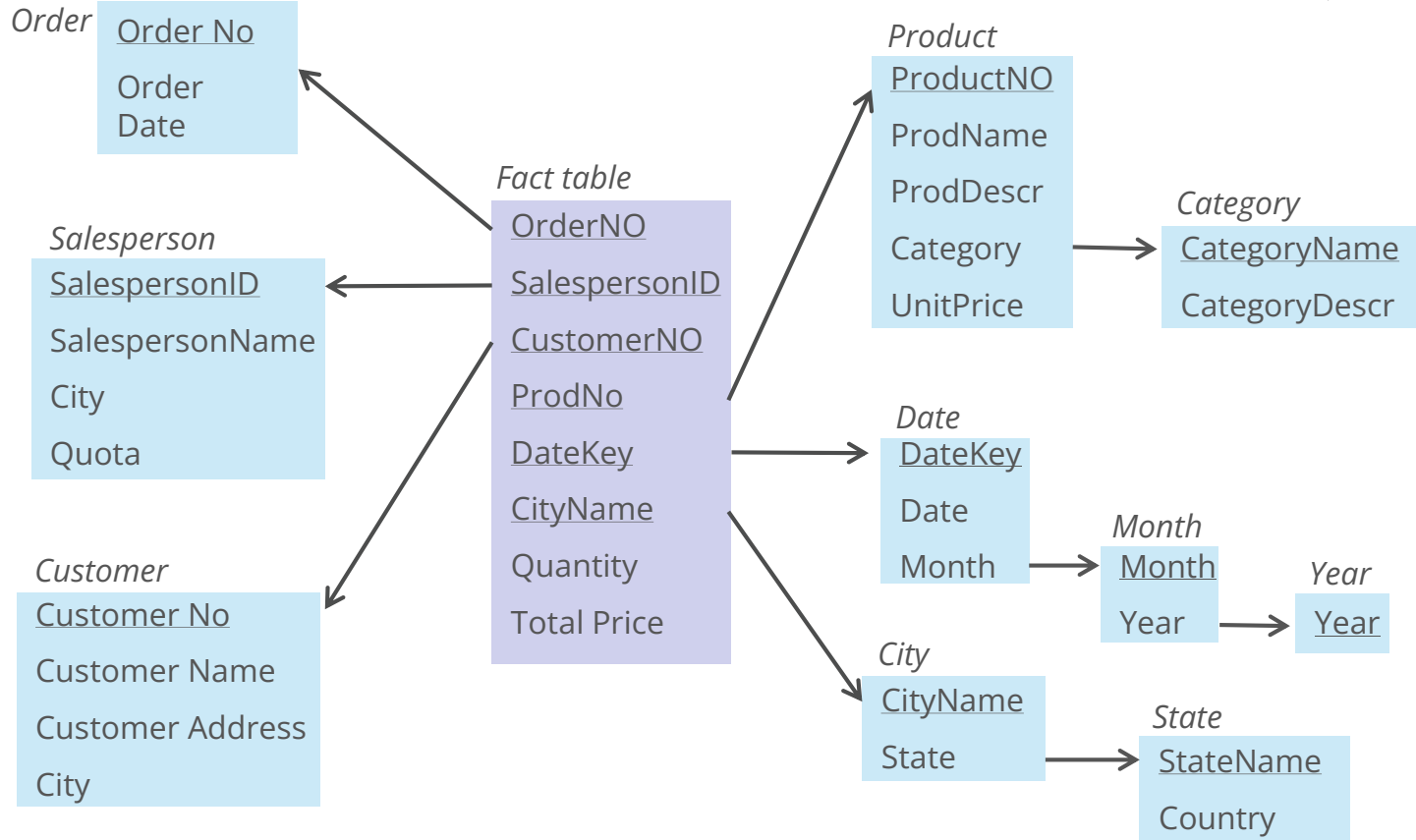
Order dimension star schema



Order dimension snowflake schema



Snowflake Schema



Main property: normalized dimension tables

- + No redundancy
- + Update-friendly
- + Inherent mapping of structural information of dimension hierarchies
- Less performance due to additional joins

Redundancy: “Resist the urge to normalize”

- Fact table size of hundreds of GBs/TBs
- Plus, index on fact table of similar size
- Dimension tables size of several hundred MBs
- Little storage savings from normalization
- Only favorable for large dimensions or many updates

Benefits Star over Snowflake

Higher query performance

- Analytic queries typically address higher aggregation levels
- Less joins because of denormalization

Data volume

- Dimension tables small in comparison to fact tables
- Additional storage required for denormalized dimension tables rather small

Changes of classification structure (meta/instance level)

- Occur rarely
- Drawback: higher change costs due to duplicate identification for denormalized dimension tables

Simplicity of structure

- Fewer dimension tables than snowflake schema
- Simpler creation of SQL queries by the OLAP-server

Combining Star & Snowflake Schema

Properties

- Some dimensions normalized
- Some dimensions denormalized

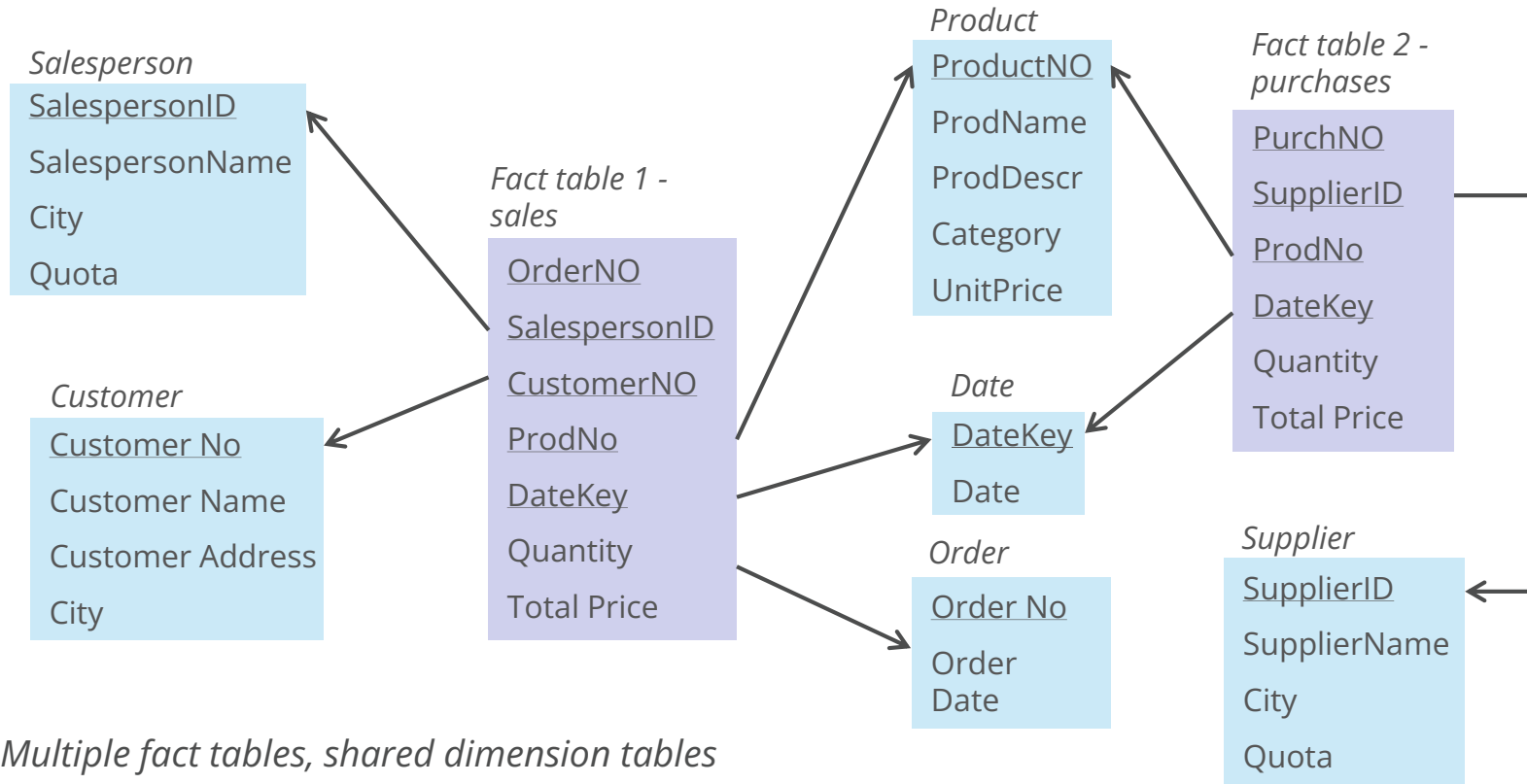
Frequency of updates

- If update frequency of a dimension is high, it makes sense to normalize this dimension to drastically reduce update costs

Number of dimension elements

- High number of finest-grain dimension elements leads to high savings through normalization
- High number of aggregation levels leads to high savings through normalization
- Multiplicative relationship

Galaxy Schema



Relational Mapping of Hierarchies

Horizontal mapping

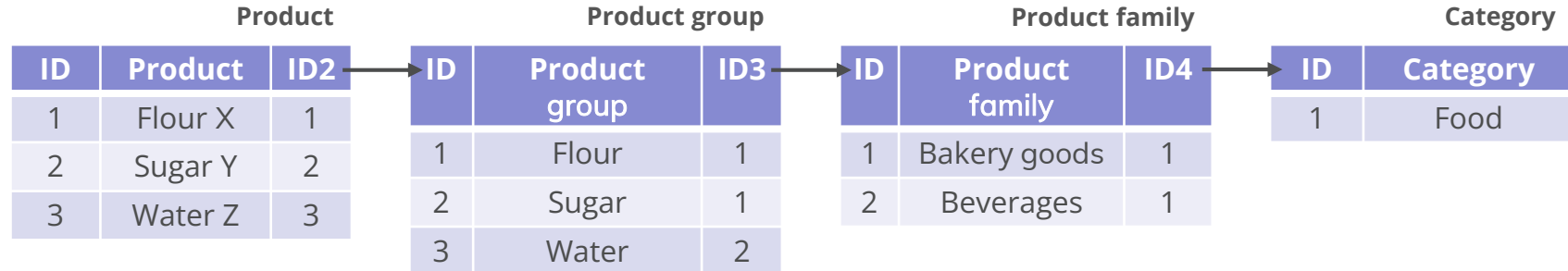
- Implicit mapping of equally ranked attributes
- Denormalized
- Ref. star schema

Relation product

ID	Product	Product group	Product family	Category
1	Flour X	Flour	Bakery goods	Food
2	Sugar Y	Sugar	Bakery goods	Food
3	Water Z	Water	Beverages	Food

Vertical Mapping

- Explicit mapping using separate relations
- Normalized
- Ref. snowflake schema



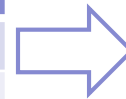
Relational Mapping of Hierarchies

Recursive, vertical mapping

- Explicit mapping of arbitrary deep hierarchies
- Combination with horizontal mapping possible
- Extension for irregular hierarchies possible
- Recursive resolution of hierarchy via a set of self-joins

Example

ID	Product	Product group	Product family	Category
1	Flour X	Flour	Bakery goods	Food
2	Sugar Y	Sugar	Bakery goods	Food
3	Water Z	Water	Beverages	Food



Relation product

ID	Product	ParentID
1	Food	NULL
2	Bakery goods	1
3	Beverages	1
4	Flour	2
5	Sugar	2
6	Water	3
7	Flour X	4
8	Sugar Y	5
9	Water Z	6

Primary/foreign key relation between dimension- & fact table

- Referential integrity: assure that every quantified part can be evaluated against the descriptive part of a multidimensional schema
- Every part of the composite primary key of the fact table has to be a foreign key to the primary key of the corresponding dimension table

SQL level: Add primary/foreign key references

- Assure referential integrity at CREATE TABLE, or later with ALTER TABLE command
- Example order dimension ORDERS

```
ALTER TABLE TPCD.ORDERS  
  ADD FOREIGN KEY ORDERS_FK1 (O_CUSTKEY) REFERENCES TPCD.CUSTOMER;
```

```
SET INTEGRITY FOR TPCD.ORDERS IMMEDIATE CHECKED;
```

Problem

- PK/FK relations can be declared using classical DB means, **BUT** declaration of functional dependencies between attributes of a dimension → requires extensions , e.g. CREATE DIMENSION (Oracle)
- Knowledge about functional dependencies is necessary to check disjointness
- Information assurances describe functional dependencies, but inside a relation (e.g. denormalized dimension tables in star schemas

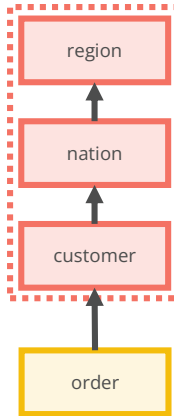
Example: Oracle SQL-dialect

```
CREATE DIMENSION ORDER_DIM
  LEVEL ORDER      IS ORDERS.ORDERKEY
  LEVEL CUSTOMER    IS ORDERS.CUSTOMERKEY
  LEVEL NATION      IS ORDERS.NATIONKEY
  LEVEL REGION      IS ORDERS.REGIONKEY

HIERARCHY ORDER_HIERARCHY (
  ORDER      CHILD OF
  CUSTOMER   CHILD OF
  NATION     CHILD OF
  REGION )

ATTRIBUTE ORDER DETERMINES
  (O_ORDERSTATUS, O_ORDERDATE,
  ...)

ATTRIBUTE CUSTOMER DETERMINES
  (C_NAME, C_ADDRESS, C_PHONE,
  ...);
```



Summary

Multidimensional Modelling

- OLAP specific modelling
- Data cube with dimensions and facts
- Operations: slice, dice, drill-down, rollup
- Summability

Relational Modelling

- MOLAP, ROLAP, HOLAP
- Snowflake / Star schema
- Representation of Hierarchies